

Programmation I : TD/TP4
(Pointeurs et Fonctions)

Exercice 1 : Donnez les valeurs de A,B,C, P1 et P2 après chaque instruction du programme suivant :

```
main()
{
    int A = 1, B = 2, C = 3, *P1, *P2;
    P1 = &A; P2 = &C;
    *P1 = (*P2)++;
    P1 = P2;
    P2 = &B;
    *P1 = *P2;
    ++*P2;
    *P1 = *P2;
    A = ++*P2**P1;
    P1 = &A;
    *P2 = *P1 /= *P2;
}
```

Exercice 2 : Ecrivez un programme qui range les éléments d'un tableau T de type float dans l'ordre inverse en utilisant uniquement le formalisme pointeur.

Exercice 3 : Ecrivez un programme qui lit un tableau A du type int au clavier et élimine toutes les occurrences de la valeur 0 dans A en tassant les éléments restants. Le programme utilisera deux pointeurs P1 et P2 pour parcourir le tableau.

Exercice 4 : Ecrire un programme qui lit deux tableaux A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A et affiche le tableau résultant. Utiliser le formalisme pointeur et l'allocation dynamique de la mémoire à chaque fois que cela est possible.

Exercice 5 : Donnez la séquence d'affichages produits par le programme suivant :

```
#include <stdio.h>
```

```
int fct ( int );
```

```
void f ( void );
```

```
int n = 10, q = 2;
```

```
main()
```

```
{ int n = 0, p = 5;
```

```
  n = fct ( p );
```

```
  printf("A : dans main, n = %d, p = %d, q = %d\n", n, p, q);
```

```
  f();
```

```
}
```

```
int fct ( int p )
```

```
{ int q;
```

```
  q = 2 * p + n;
```

```
  printf("B : dans fct, n = %d, p = %d, q = %d\n", n, p, q);
```

```
  return q;
```

```
}
```

```
void f ( void )
```

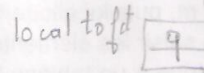
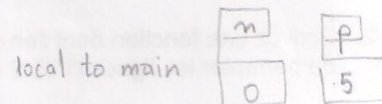
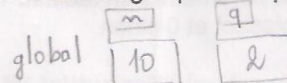
```
{ int p = q * n;
```

```
  printf("C : dans f, n = %d, p = %d, q = %d\n", n, p, q);
```

```
}
```

Exercice 6 : Le nombre de combinaisons de k éléments parmi n est donné par : $C_n^k = \frac{n!}{k!(n-k)!}$

Ecrivez une fonction qui permet de calculer les coefficients binomiaux C_n^k .



B : dans fct, n = 10, p = 5, q = 20 (1)



C : dans f, n = 10, p = 20, q = 2 (3)

Le triangle de Pascal, est un arrangement géométrique des coefficients binomiaux dans un triangle.

À la ligne i et à la colonne j ($0 \leq j \leq i$) est placé le coefficient binomial C_i^j :

$$\begin{array}{c} C_0^0 \\ C_1^0 \quad C_1^1 \\ C_2^0 \quad C_2^1 \quad C_2^2 \\ \dots \end{array}$$

Ecrivez un programme qui construit et affiche le triangle de Pascal.

Exercice 7 : Qu'affiche le programme suivant à l'écran ?

```
#include<stdio.h>
int mystere(int a,int *b)
{ a=*b*2;
  *b=*b+a;
  return (*b+1);
}
```

```
int main(){
  int x=1,y=1,z=1;
  z=mystere(x,&y);
  printf("x=%d,y=%d,z=%d\n",x,y,z);
  x=mystere(x,&y);
  printf("x=%d,y=%d,z=%d\n",x,y,z);
}
```

Exercice 8 : Le pgcd de deux entiers positifs est le plus grand entier positif qui divise les deux.

Quand on étudie les propriétés du pgcd on voit que :

$\text{pgcd}(x, y) = \text{pgcd}(x - y, y)$ si $x > y$

$\text{pgcd}(x, y) = \text{pgcd}(x, y - x)$ si $x < y$

$\text{pgcd}(x, 0) = \text{pgcd}(0, x) = x$

1. Ecrire une fonction récursive pour le calcul du pgcd.
2. Ecrire un programme C qui fait appel à la fonction pgcd.

Exercice 9 : Dans tout cet exercice, utilisez le formalisme pointeur et l'allocation dynamique de la mémoire à chaque fois que cela est possible.

1. Ecrivez une fonction qui prend en paramètres un tableau T de type int et sa taille n et retourne 1 si le tableau est trié par ordre croissant et 0 sinon.
2. Ecrivez une fonction dont l'en-tête est **void Permut(int *M, int n, int m, int l1, int l2)** qui permet de permuter les lignes l1 et l2 d'une matrice M de dimensions n et m.
3. Ecrivez la fonction principale main() qui :
 - a) Déclare d'abord une matrice A sous forme de pointeur (int *A), demande à l'utilisateur de saisir ses dimensions n et m, puis lui alloue la mémoire dynamiquement.
 - b) Demande à l'utilisateur de saisir les éléments de la matrice A puis l'affiche.
 - c) Construit à partir de la matrice A un tableau V : V[i] est égal à 1 si la ligne i de A est triée et égal à 0 sinon (utiliser la fonction de la question 1).
 - d) Arrange la matrice A de telle sorte que les lignes non triées soient au début de la matrice et les lignes triées à la fin (utiliser la fonction de la question 2).

Exemple : $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 1 & 4 \\ 8 & 2 & 1 \end{pmatrix} \Rightarrow V = (1, 1, 0, 0) \Rightarrow A = \begin{pmatrix} 8 & 2 & 1 \\ 7 & 1 & 4 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$